



# A novel learning-based feature recognition method using multiple sectional view representation

Peizhi Shi<sup>1</sup> · Qunfen Qi<sup>1</sup> · Yuchu Qin<sup>1</sup> · Paul J. Scott<sup>1</sup> · Xiangqian Jiang<sup>1</sup>

Received: 19 September 2019 / Accepted: 2 January 2020 / Published online: 22 January 2020  
© The Author(s) 2020

## Abstract

In computer-aided design (CAD) and process planning (CAPP), feature recognition is an essential task which identifies the feature type of a 3D model for computer-aided manufacturing (CAM). In general, traditional rule-based feature recognition methods are computationally expensive, and dependent on surface or feature types. In addition, it is quite challenging to design proper rules to recognise intersecting features. Recently, a learning-based method, named FeatureNet, has been proposed for both single and multi-feature recognition. This is a general purpose algorithm which is capable of dealing with any type of features and surfaces. However, thousands of annotated training samples for each feature are required for training to achieve a high single feature recognition accuracy, which makes this technique difficult to use in practice. In addition, experimental results suggest that multi-feature recognition part in this approach works very well on intersecting features with small overlapping areas, but may fail when recognising highly intersecting features. To address the above issues, a deep learning framework based on multiple sectional view (MSV) representation named MsvNet is proposed for feature recognition. In the MsvNet, MSVs of a 3D model are collected as the input of the deep network, and the information achieved from different views are combined via the neural network for recognition. In addition to MSV representation, some advanced learning strategies (e.g. transfer learning, data augmentation) are also employed to minimise the number of training samples and training time. For multi-feature recognition, a novel view-based feature segmentation and recognition algorithm is presented. Experimental results demonstrate that the proposed approach can achieve the state-of-the-art single feature performance on the FeatureNet dataset with only a very small number of training samples (e.g. 8–32 samples for each feature), and outperforms the state-of-the-art learning-based multi-feature recognition method in terms of recognition performances.

**Keywords** Feature recognition · Deep learning · Multiple sectional views · Transfer learning · Data augmentation

## Introduction

Computer-aided process planning (CAPP) is an important phase which aims to generate a set of manufacturing operations for a product according to its computer-aided design (CAD) data. Typical CAD models only contain pure geometry and topology information, which is considered as low-level information, e.g. faces, edges and vertices. Therefore, the first major task in the CAPP is called feature recognition, that is to identify high-level machining features (e.g. slots, holes and steps) of a CAD model from its intrinsic

geometry and topology information. According to the identified feature type, a sequence of instructions can then be generated to be used in computer-aided manufacturing (CAM). However, how to recognise the feature type in an effective and efficient manner remains a challenging task (Han et al. 2000; Babic et al. 2008; Gao 1998; Verma and Rajotia 2010; Xu 2009).

In digital manufacturing, feature recognition should be preferably performed in an automatic manner. A number of research has been conducted to automatically recognise the feature types of 3D CAD models with regular surfaces (e.g. planes, cylinders and conic) or freeform surfaces (Sundararajan and Wright 2004; Sunil and Pande 2008; Lingam et al. 2017). In general, these methods can be carried out through either a rule-based (Babic et al. 2008) or learning-based approach (Babić et al. 2011). In rule-based methods, rule designers need to fully understand the

✉ Qunfen Qi  
q.qi@hud.ac.uk

<sup>1</sup> EPSRC Future Advanced Metrology Hub, School of Computing and Engineering, University of Huddersfield, Huddersfield HD1 3DH, UK

natures of different types of surfaces and structures, and then encode their knowledge into heuristic rules for feature recognition. Those heuristic rules are ad-hoc, and dependent on the feature and surface types (Henderson et al. 1994). Different rules/algorithms might be required for different types of features or surfaces, where great efforts and inside knowledge are required. In addition, most rule-based approaches adopted a search algorithm, which is computationally intensive. In contrast to rule-based methods, learning-based approaches construct a feature recogniser via machine learning from human annotated dataset. Recently, Zhang et al. (2018) proposed a promising learning-based feature recognition approach to both single and multi-feature recognition, which is capable of addressing the issues raised in the existing methods. In this approach, a 3D convolutional neural network (CNN) was employed to recognise the feature type according to a 3D voxel model. This is a general purpose algorithm, where rule designers do not need to encode their knowledge into rules. However, thousands of training samples per class are required to train a reliable classifier, which also incurs great human efforts. In addition, experimental results suggest that the multi-feature recognition part in this approach works very well on intersecting features with small overlapping areas, but may fail when recognising highly intersecting features.

Motivated by the multi-view neural network (Su et al. 2015), where multiple views of a 3D model are combined together via the neural network and result in a better 3D object classification performance, a novel view-based approach is proposed in this paper to address the aforementioned issues. In the proposed approach to single feature recognition, a novel representation based on multiple sectional views is employed to represent a 3D model. A deep neural network named MsvNet is implemented based on this view-based representation. Then, transfer learning and fine-tuning are first employed to initialise the parameters in the neural network, while multiple view training is adopted to fully train the MsvNet. During the training process, data augmentation is used to enhance the learning performances. For multi-feature recognition, a novel view-based feature segmentation is adopted to capture all possible features from 2D views of a 3D model. Then, the MsvNet trained previously is employed to recognise those segmented features. Experimental results suggest that (1) the MsvNet can achieve the state-of-the-art single feature recognition performance on the benchmark dataset provided by Zhang et al. (2018) with only a small number of training samples (e.g. 8–32 samples for each feature); (2) the training strategies employed in the MsvNet (e.g. MSV-based representation, transfer learning and data augmentation) increase the recognition performance by a considerable amount; (3) the proposed view-based multi-feature segmentation and recognition method outperforms

the state-of-the-art learning-based method on the benchmark multi-feature dataset constructed in this paper.

The contributions of the paper are as follows: (1) A novel learning-based single feature recognition method which achieves good performance with only a few training samples is proposed. (2) A novel multi-feature segmentation and recognition method which outperforms the state-of-the-art learning-based multi-feature recognition method is presented. (3) A thorough evaluation of the proposed approach is presented.

The rest of the paper is organised as follows. Section “Related work” reviews the related work that motivates the proposed approach. Section “Single feature recognition” describes the MsvNet for single feature recognition in detail. Section “Multi-feature recognition” discusses how to employ MsvNet for multi-feature recognition. Section “Results for single feature recognition” provides the results for single feature recognition in detail. Section “Results for multi-feature recognition” further examines the benefits of the proposed multi-feature recognition method. Section “Conclusion” summarises the main contributions of the proposed approach and discusses possible future work.

## Related work

As described in section “Introduction”, two types of methodologies are adopted for feature recognition: rule- and learning-based approaches. This section reviews a number of rule- and learning-based approaches to feature recognition.

There are a variety of rule-based approaches, e.g. graph-based method (Joshi and Chang 1988; Huang and Yip-Hoi 2002; Lockett and Guenov 2005; Kao 1993; Li et al. 2010; Xu et al. 2015; Campana and Mele 2018), hint-based method (Vandenbrande and Requicha 1993; Han and Requicha 1998; Han et al. 2001b; Han and Requicha 1997), cell-based approach (Woo 2003), ontology-based method (Wang and Yu 2014; Zhang et al. 2017), STEP-based approach (Mokhtar et al. 2009; Venu and Komma 2017; Venu et al. 2018; Al-wswasi and Ivanov 2019; Han et al. 2001a; Ong et al. 2003; Kannan and Shunmugam 2009a, b; Dipper et al. 2011; Mokhtar and Xu 2011), planning approach (Marchetta and Forradellas 2010), tree-based method (Li et al. 2002; Sung et al. 2001), hybrid approach (Gao and Shah 1998; Rahmani and Arezoo 2007; Rameshbabu and Shunmugam 2009; Hayasi and Asiabanpour 2009), and others (Zhang et al. 2014; Harik et al. 2017). A typical rule-based feature recognition approach is the graph-based method (Joshi and Chang 1988), in which a feature recognition problem is formulated as a graph matching problem. In this approach, a boundary representation (B-rep) is employed to represent a 3D model. Then, an adjacency attributed graph (AAG) is captured from the model. In an AAG, a node represents a face appeared

in the 3D model, while an edge represents the adjacency relation between two faces. Once the AAG is constructed, a sub-graph matching algorithm is employed to find a sub-graph in the AAG which is isomorphic to a pre-defined graph in a graph database. If an isomorphism exists between the two graphs, the feature type of the 3D model can be identified accordingly. Both graph-based and other rule-based approaches has the following limitations: (1) In general, most rule-based approaches employ a search algorithm (e.g. sub-graph matching algorithm) for feature recognition, which is computationally expensive (Babic et al. 2008). (2) To design a rule-based approach, a rule designer needs to fully understand the natures of different types of surfaces and structures, design specific rules for each feature or surface type. When recognising a variety types of features, great human efforts are required. For instance, the typical graph-based approach is only suitable for negative, polyhedral 3D models (Babic et al. 2008). To recognise models with complicated structures (e.g. 3D models with curved faces), advanced techniques, e.g. multi-attributed adjacency graph (Venuvinod and Wong 1995), are required. Thus, a general purpose feature recognition algorithm that is suitable for all types of surface and feature is needed. (3) Recognising intersecting features is a challenging issue, which becomes a major bottleneck for feature recognition (Han et al. 2000). (4) Existing rule-based approaches are not capable of learning new rules from new features (Sunil and Pande 2009).

Using machine learning techniques for feature recognition has a relatively long history. Hwang (1992) presented an approach to recognising features according to B-rep 3D models via a single layer perceptron. In this approach, a 8D score vector achieved from a B-rep model was employed as the input of the network. Onwubolu (1999) implemented a multi-layer feed-forward network for feature recognition, where the input was a 9D score vector. One issue raised in this approach is that it is only capable of recognising a small number of features. Also, the number of faces appeared in the 3D model should be fixed in this approach. Sunil and Pande (2009) proposed a learning-based method which was capable of recognising a wide range of features. In this approach, the topology and geometry information of a feature was encoded in a 12D score vector achieved from a B-rep model. Then, the 12D score vector was employed as the input of the neural network. Above learning-based approaches were proposed based on the boundary representation, which might not be generalised to other representations easily (Zhang et al. 2018). In addition, these approaches largely rely on well-designed score vectors as inputs. Designing a reliable score vector for feature recognition will impose burdens on human designers. Furthermore, these approaches are not capable of recognising overlapping features. Most existing learning-based approaches (Prabhakar 1990; Nezis and Vosniakos 1997; Ding and Yue 2004; Öztürk and Öztürk 2004;

Brousseau et al. 2008; Öztürk and Öztürk 2001) also face the similar issues. To tackle the issues raised in existing rule- and learning-based methods, Zhang et al. (2018) presented a novel approach named FeatureNet to handle single and multi-feature recognition problems. In this approach, a large dataset that consists of 144,000 3D models of 24 types of features was constructed. For single feature recognition, a 3D CNN was employed to learn a mapping from a 3D voxel model onto its feature type. In multi-feature recognition, a 3D model segmentation method named watershed algorithm is employed for multi-feature segmentation and recognition. As a general purpose algorithm, designers do not need to design rules or input score vectors for different features/surfaces. However, thousands of training samples per class are required to achieve a reliable feature recogniser. In practice, a great human efforts and time are required to attain and annotate a large number of training data. In addition, experimental results suggest that watershed algorithm works very well on intersecting features with small overlapping areas, but may fail when recognising highly intersecting features.

## Single feature recognition

This section first provides the motivation for undertaking this research, and offers an overview of the proposed method for single feature recognition. Then, the representation employed in this method and machine learning techniques are presented in details.

### Overview

As “Related work” section has specified, existing feature recognition methods suffer from a number of issues. Motivated by these research gaps, this section formulates the main research problem as: how to implement a general purpose feature recogniser with only a few training samples via machine learning techniques. To tackle this problem, a number of machine learning techniques are studied and applied.

Feature recognition can be regarded as a 3D object classification problem, which is a popular research topic in the area of computer vision. To enable effective object classification, two types of methods have been proposed: voxel- and view-based approaches. In the voxel-based approach (Qi et al. 2016; Zhang et al. 2018), a model was represented as a 3D occupancy grid, and used as the input of the deep neural network. In the view-based approach (Su et al. 2015), a number of 2D view images were collected from a 3D model, and adopted as input of the network. Experimental results suggested that the view-based representation outperformed the voxel-based representation in terms of classification accuracy under the same experimental settings (Su et al. 2015, 2018; Qi et al. 2016).



**Fig. 1** System diagram

In a real-world application, the number of training samples might be limited, which can result in a poor learning performance. In addition to the view-based representation, the other two methods that could minimise the number of required training samples are transfer learning (Pan and Yang 2009) and data augmentation. Transfer learning aims to employ the knowledge gained from one problem to solve another different but related problem. In practice, transfer learning reuses a pre-trained model on another dataset to initialise the parameters in the current network. Data augmentation aims to increase the diversity of the training set without collecting new samples. This technique was first designed for 2D images, but can also be applied to 3D models. These two simple strategies enable users to train a reliable classifier with only a small number of training samples.

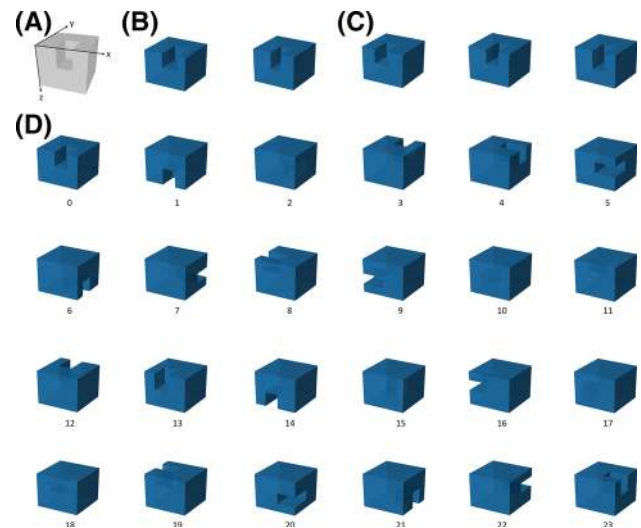
The above description implies that view-based representation, transfer learning and data augmentation would allow for addressing the research problem effectively. To this end, a deep learning approach to feature recognition named MsvNet is proposed. The system diagram is illustrated in Fig. 1. It is observed from this figure that the system consists of three components: data processing, training stage 1 and 2. At the data processing stage, data augmentation technique is employed to increase the diversity of the training set, and multiple sectional views of training samples are collected as the input of deep neural network. At the training stage 1, a deep neural network is created, and two learning strategies named transfer learning (or pre-training) and fine-tuning are employed to initialise the parameters in the neural network. A well-initialised deep network is then constructed ready for further training. At the stage 2, information achieved from different views are combined via the neural network constructed at the previous stage for training. This section discusses above steps in detail.

### Data pre-processing

In this method, data augmentation and view-based representation are adopted for pre-processing to achieve a classifier with good generalisation performance.

### Data augmentation

This approach employs three data augmentation strategies, which are illustrated in Fig. 2.



**Fig. 2** Data augmentation process: **a** original 3D model, **b** randomly resized models, **c** features moved along  $x$ -,  $y$ - and  $z$ -axis, and **d** rotated models

- In the first data augmentation strategy, the 3D model is resized by a random scale factor. If the resized model is smaller than the original model, edge padding is applied to the resized model. If the resized model is larger than the original model, random cropping is applied to the resized model. Above operations could make the size of new 3D model equals to the original size.
- In the second strategy, the 3D model is moved along  $x$ -,  $y$ - or  $z$ -axis by a random amount.
- A 3D model could be rotated along the  $x$ -,  $y$ - or  $z$ -axis, which produces 24 orientations. In the final strategy, an orientation is randomly selected from 24 orientations, and the 3D model with this orientation is employed during training. The rotation operators are illustrated in Table 1.

One thing that needs to be noticed is that data augmentation may change the topology of features. Suppose that the first data augmentation strategy is applied to a blind hole, after adopting the resizing and random crop to this feature, it is very likely to achieve another blind hole with the same topology structure, but with a different size. This is an expected result, since this technique could largely increase the diversity of the training dataset. In some situations, however, some incorrect features (e.g. a half blind hole, a through hole) can also be constructed. These incorrect features can be regarded as noisy data. To minimise this effect, this approach employs only a small resize scale during data augmentation. According to the calculation, around 94.89% correct yet useful features could be achieved when applying data augmentation to a feature, while only 5.11% incorrect noisy features will be produced. In the area of computer vision, the noises introduced by data augmentation cannot be totally

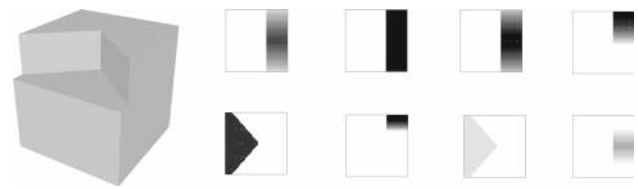
**Table 1** Model rotation

|    | Rotation operator  |
|----|--|
| 0  | Original model   |
| 1  | Rotate the model 90° anticlockwise around the $x$ axis   |
| 2  | Rotate the model 180° around the $x$ axis  |
| 3  | Rotate the model 90° clockwise around the $x$ axis   |
| 4  | Rotate the model 90° clockwise around the $z$ axis   |
| 5  | Rotate the model 90° clockwise around the $z$ axis, then rotate it 90° anticlockwise around the $x$ axis     |
| 6  | Rotate the model 90° clockwise around the $z$ axis, then rotate it 180° around the $x$ axis                  |
| 7  | Rotate the model 90° clockwise around the $z$ axis, then rotate it 90° clockwise around the $x$ axis         |
| 8  | Rotate the model 90° anticlockwise around the $z$ axis   |
| 9  | Rotate the model 90° anticlockwise around the $z$ axis, then rotate it 90° anticlockwise around the $x$ axis |
| 10 | Rotate the model 90° anticlockwise around the $z$ axis, then rotate it 180° around the $x$ axis              |
| 11 | Rotate the model 90° anticlockwise around the $z$ axis, then rotate it 90° clockwise around the $x$ axis     |
| 12 | Rotate the model 180° around the $z$ axis  |
| 13 | Rotate the model 180° around the $z$ axis, then rotate it 90° anticlockwise around the $x$ axis              |
| 14 | Rotate the model 180° around the $z$ axis, then rotate it 180° around the $x$ axis                           |
| 15 | Rotate the model 180° around the $z$ axis, then rotate it 90° clockwise around the $x$ axis                  |
| 16 | Rotate the model 90° clockwise around the $y$ axis   |
| 17 | Rotate the model 90° clockwise around the $y$ axis, then rotate it 90° anticlockwise around the $x$ axis     |
| 18 | Rotate the model 90° clockwise around the $y$ axis, then rotate it 180° around the $x$ axis                  |
| 19 | Rotate the model 90° clockwise around the $y$ axis, then rotate it 90° clockwise around the $x$ axis         |
| 20 | Rotate the model 90° anticlockwise around the $y$ axis   |
| 21 | Rotate the model 90° anticlockwise around the $y$ axis, then rotate it 90° anticlockwise around the $x$ axis |
| 22 | Rotate the model 90° anticlockwise around the $y$ axis, then rotate it 180° around the $x$ axis              |
| 23 | Rotate the model 90° anticlockwise around the $y$ axis, then rotate it 90° clockwise around the $x$ axis     |

avoided, but will not significantly affect the learning performances (Wang et al. 2018).

### Multiple sectional view representation

In the view-based representation (Su et al. 2015), several virtual cameras are set up around a 3D model, and used to collect

**Fig. 3** A 2-sides through step with a number of sectional views

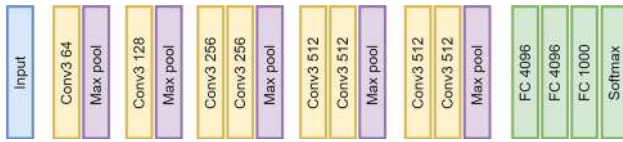
a number of rendered 2D images from different viewpoints. These 2D images are employed to represent the 3D object.

This paper presents a multiple sectional view representation for feature recognition. In this representation, a 3D object is cut via a number of randomly placed cutting planes which are vertical to  $x$ -,  $y$ - or  $z$ -axis, and corresponding 2D sectional view images are captured to represent this 3D model, as shown in Fig. 3. In comparison to the typical view-based representation (Su et al. 2015), the proposed representation contains the hidden information inside the 3D object, and can be achieved efficiently.

### Stage 1: pre-training and fine-tuning

In addition to the representation, another essential issue in deep learning is how to construct and initialise the neural network. In general, a proper network architecture with well-initialised parameters could produce a good learning result.

To attain the above goal, transfer learning is applied in this paper. This method aims to utilise the knowledge learned from a source task to help improve the target task performance. The transferred knowledge is normally encoded in a machine learning model trained on a dataset for the source task, and is reused to initialise the parameters (e.g. weights, biases of neural networks) in the target model. This simple strategy enables users to train a reliable classifier with only a small number of training samples. Transfer learning normally works on an assumption that the source and target tasks are different but related (Pan and Yang 2009). In this paper, the target task is to employ 2D views for object classification, which can be regarded as an image classification problem. To this end, this system adopts a VGGNet-11 (Simonyan and Zisserman 2014) as it is an effective deep convolutional network which is widely used for image classification. Then, a pre-trained VGG-11 model on the ImageNet dataset is adopted to initialise the model parameters (e.g. weights, biases) in the current network as ImageNet is a huge benchmark dataset for image classification. It is observed from the above description that source (ImageNet image classification) and target (2D view-based object classification) tasks are related, which can ensure that this transfer is suitable. The validation of transfer learning is presented in “Effects of different learning strategies” section.



**Fig. 4** VGGNet-11 architecture

Figure 4 illustrates the architecture of the VGGNet-11 deep convolutional network. The input of a VGGNet-11 is a 3-channel RGB image with a shape  $3 \times 224 \times 224$ . It is observed from the figure that this network consists of eight convolutional layers, where a number of kernels are applied to the input to calculate the output of neurons. A 2D convolution operation is defined as follow:

$$z_{i,j,k} = \sum_{l,m,n} g_{l,j+m-1,k+n-1} k_{i,l,m,n} \quad (1)$$

where  $k$  refers to the kernel,  $k_{i,l,m,n}$  refers to the strength between the input channel  $l$  and the output channel  $i$  at the location  $(m, n)$ ;  $g$  denotes the input; and  $z_{i,j,k}$  refers to the output on the channel  $i$  at the location  $(j, k)$ . In the VGGNet-11, all kernel sizes are set to  $3 \times 3$ , and the number of output channels for eight layers are set to 64, 128, 256, 256, 512, 512, 512, and 512, respectively. After each convolutional layer, an activation function named rectified linear units (ReLU) is employed to determine whether the current neuron should be activated or not. The ReLU operation is defined as follow:

$$o_{i,j,k} = \max(0, z_{i,j,k}) \quad (2)$$

where  $o_{i,j,k}$  refers to the output on the channel  $i$  at the location  $(j, k)$  after the ReLU layer. It is also observed from the figure that three fully connected layers and one softmax layer are employed. The output of each neuron in a fully connected layer can be calculated as:

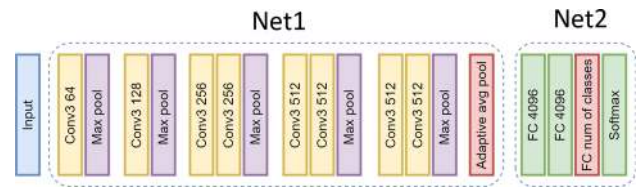
$$u = \max\left(0, \sum_i w_i x_i + b\right) \quad (3)$$

where  $x$  refers to the input of the neuron; while  $w$  and  $b$  refer to the weights and bias in the neuron, respectively. The  $i$ th output in the softmax layer is calculated as:

$$\hat{y}_i = \frac{e^{u_i}}{\sum_j e^{u_j}} \quad (4)$$

where  $u_i$  refers to the output of the  $i$ th neuron in the final fully connected layer. In the VGGNet-11, the final output is a 1000D vector as the ImageNet dataset contains 1000 different image categories.

The typical VGGNet-11 takes an image of a high resolution ( $3 \times 224 \times 224$ ) as input, which is computationally



**Fig. 5** Network architecture employed at the stage 1 (the adapted layers are highlighted with red color) (Color figure online)

expensive in feature recognition. To handle an image with a different input size (e.g.  $3 \times 64 \times 64$ ), an adaptive average pooling layer is added before the fully connected layers. In addition, the final fully connected layer of the network is replaced with another fully connected layer in which the number of neurons equals to the number of feature types. The weights in the new layer are randomly initialised, while the biases are set to zero. The adapted network architecture is illustrated in Fig. 5. In the proposed method, convolutional layers and pooling layers are denoted as Net1, while fully connected layers and softmax layer are denoted as Net2.

Once the network architecture is determined and the parameters are well initialised, another task suggested by Su et al. (2015) is to further fine-tune the parameters in the network using the domain dataset (i.e. dataset for feature recognition). At this stage, the network is trained as a single-image recognition problem, where the input of the network is a single sectional view of a 3D model, and the output is the 3D model's feature type. The cross-entropy loss is employed for training, which is defined as follow:

$$L = -\frac{1}{c} \sum_i y_i \log \hat{y}_i \quad (5)$$

where  $c$  refers to the number of classes,  $y$  is the ground-truth output vector, and  $\hat{y}$  is the predicted output vector. Adam optimizer is adopted to minimise the loss function as it converges to the optimal solution quickly.

## Stage 2: multiple view training

Once a well-initialised deep network is achieved, the next task is to fully train the network by using multiple sectional views. The input of the network is a collection of multiple sectional view images of a 3D model, while the output is the 3D model's feature type. Thus, an essential issue is how to combine the information achieved from different views for feature recognition. Figure 6 shows the multiple view training method, which is capable of dealing with this issue. As shown in this figure, a view pooling layer (Su et al. 2015) is added before the Net2. In this pooling layer, an element-wise max operation is employed to combine all views into a single view. This operation is illustrated as follow:

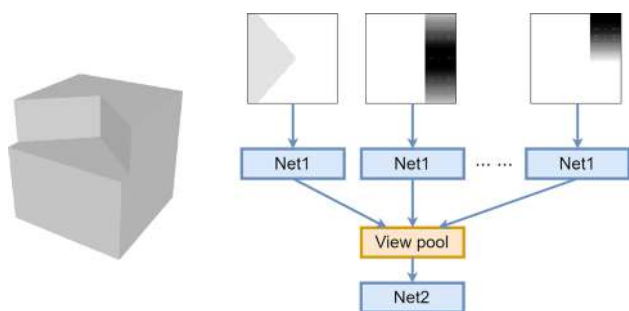


Fig. 6 Multiple view training

$$v_{i,j,k} = \max(o_{i,j,k}^1, o_{i,j,k}^2, \dots) \tag{6}$$

where  $o_{i,j,k}^1$  refers to the output of Net1 for the first view on the channel  $i$  at the location  $(j, k)$ ,  $v_{i,j,k}$  refers to the output of the view pooling layer. During the training/testing, multiple sectional views of a 3D model are passed through the Net1, combined together by the view pooling layer, and sent to the Net2 for recognition. All the networks in Net1 employ the same parameters. The cross-entropy loss and Adam optimizer are also adopted for training at this stage.

### Multi-feature recognition

A CAD model normally consists of multiple intersecting features instead of a single feature. This section presents a novel method by applying the MsvNet described in “Single feature recognition” section to multi-feature recognition. This section first identifies the issue, raised in the state-of-the-art learning-based multi-feature recognition method, which motivates the proposed approach, and offers an overview of the proposed method. Then, a novel view-based feature segmentation and recognition algorithm is presented in details.

#### Overview

As “Related work” section has specified, the state-of-the-art learning-based multi-feature recognition method (Zhang et al. 2018) employs watershed segmentation algorithm to split intersecting features appeared in a 3D model into separated ones, and recognises each of them separately. One issue raised in watershed segmentation algorithm is that it works very well on intersecting features with small overlapping areas, but may fail when recognising highly intersecting features. Figure 7 illustrates three 3D models, where each of them consists of two overlapping blind holes. The 3D model in the Fig. 7a contains a small overlapping area, the model in the Fig. 7b contains a larger overlapping area, the model in the Fig. 7c contains the largest overlapping area.

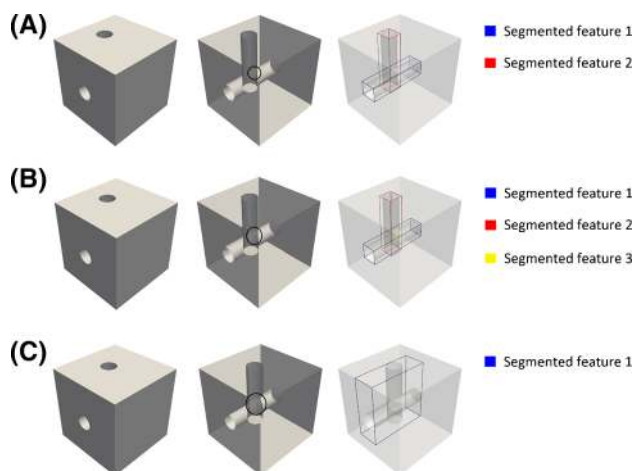


Fig. 7 Three multi-feature models with different degree of overlap, and the segmentation results achieved from the watershed algorithm. The overlapped areas are highlighted with black circles

When applying watershed algorithm<sup>1</sup> to the highly intersecting cases in Fig. 7b, c, it is observed that that watershed algorithm fails to identify the features correctly, which will lead to an inaccurate recognition result. Motivated by this research gap, this section formulates the main research problem as: how to effectively segment and recognise multiple features appeared in a 3D model.

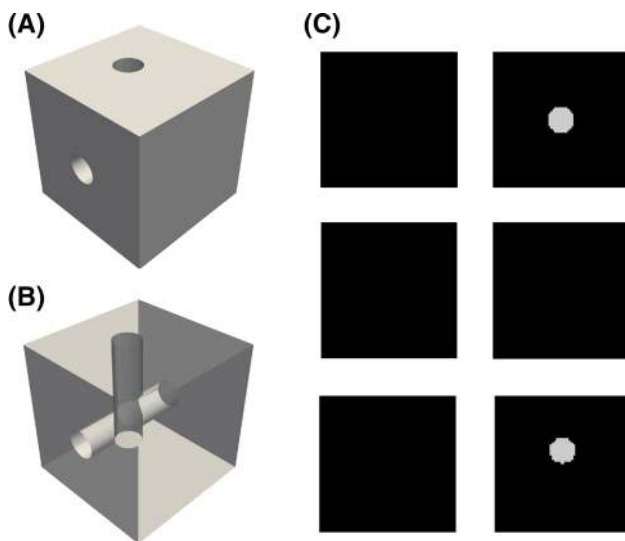
Feature segmentation in a 3D space is a difficult task, and becomes a bottleneck for multi-feature recognition. However, the authors observed that it is easier to identify multiple 3D intersecting features from 2D views. Figure 8 illustrates a 3D model with two overlapping blind holes and its six view images<sup>2</sup> taken from six directions. From these view images, it is easier to know that there are two features on the surfaces of the cube.<sup>3</sup> Then, two features can be separated and employed for further recognition. This observation suggests that a view-based method allows for addressing the multi-feature segmentation and recognition issues effectively.

To carry out the aforementioned idea, however, a number of non-trivial issues need to be tackled. In general, highly intersecting features normally consists of complicated structures, which might result in a complicated view image with overlapping features on it. It is challenging to find all possible feature locations appeared in a view image without losing any (or too much) features. To tackle this problem, region proposal algorithm for object recognition and detection in the

<sup>1</sup> The watershed algorithm with the default setting is employed in this experiment. The source code that produces the reported results is available online: <https://github.com/PeizhiShi/MsvNet>.

<sup>2</sup> In the feature segmentation, views instead of sectional views are employed here.

<sup>3</sup> This is only a simple example to introduce the view-based segmentation concept. More complicated case can be found in the following section.



**Fig. 8** A multi-feature model and its six view images



**Fig. 9** Multi-feature recognition diagram

area of computer vision appears to be a suitable method as it could capture all possible object/feature locations appeared in an image. Although the use of region proposal algorithm leads to a set of all (or most) possible feature locations, a large amount of redundant features (e.g. a hole appeared twice) and wrongly segmented features (e.g. a half triangular pocket) will also appear in this feature proposal set. To tackle this issue, a proposal selection algorithm is adopted to choose the correct results from the proposal set.

As depicted in Fig. 9, the proposed multi-feature recognition process consists of three stages: feature segmentation (to achieve all possible features appeared in a 3D model based on its view images), feature recognition (to recognise the features achieved from the previous stage via the MsvNet) and proposal selection (to achieve final recognition results). The main techniques used in each stage are presented in the following sections.

### View-based feature segmentation and recognition

As described previously, the view-based feature segmentation technique aims to achieve all possible features appeared in a 3D model based on its view images. To attain this goal, selective search (Uijlings et al. 2013), a popular yet effective region proposal algorithm in the area of computer vision, is employed as it is capable of finding all possible object/feature locations based on an image.

### Algorithm 1: Selective search algorithm

---

**Input:** an image  
**Output:** a set of object region proposals  $R$   
 Get initial regions  $R = \{r_1, \dots, r_n\}$  via segmentation  
 Initialise similarity set  $S = \emptyset$   
**foreach** connected region pair  $(r_i, r_j)$  **do**  
 | Calculate the similarity  $s(r_i, r_j)$   
 |  $S = S \cup s(r_i, r_j)$   
**end**  
**while**  $S$  contains connected regions **do**  
 | Get highest similarity score  $s(r_i, r_j)$  from  $S$   
 | Merge  $r_i$  and  $r_j$  together to form a new region  $r_l$   
 | Remove similarities regarding  $r_i$  and  $r_j$  from  $S$   
 | Achieve similarity between  $r_l$  and its neighbours  
 | Put similarities regarding  $r_l$  into  $S$  and  $R$   
**end**  
**return**  $R$

---

In the selective search algorithm (Uijlings et al. 2013), the input view image is first separated into several small regions via Felzenszwalb segmentation algorithm (Felzenszwalb and Huttenlocher 2004). Then, the similarities between all neighbouring regions are calculated. Finally, a greedy hierarchical algorithm is employed to combine regions with high similarity together. This process iterates until all connected regions are merged together. This algorithm is summarised in Algorithm 1.

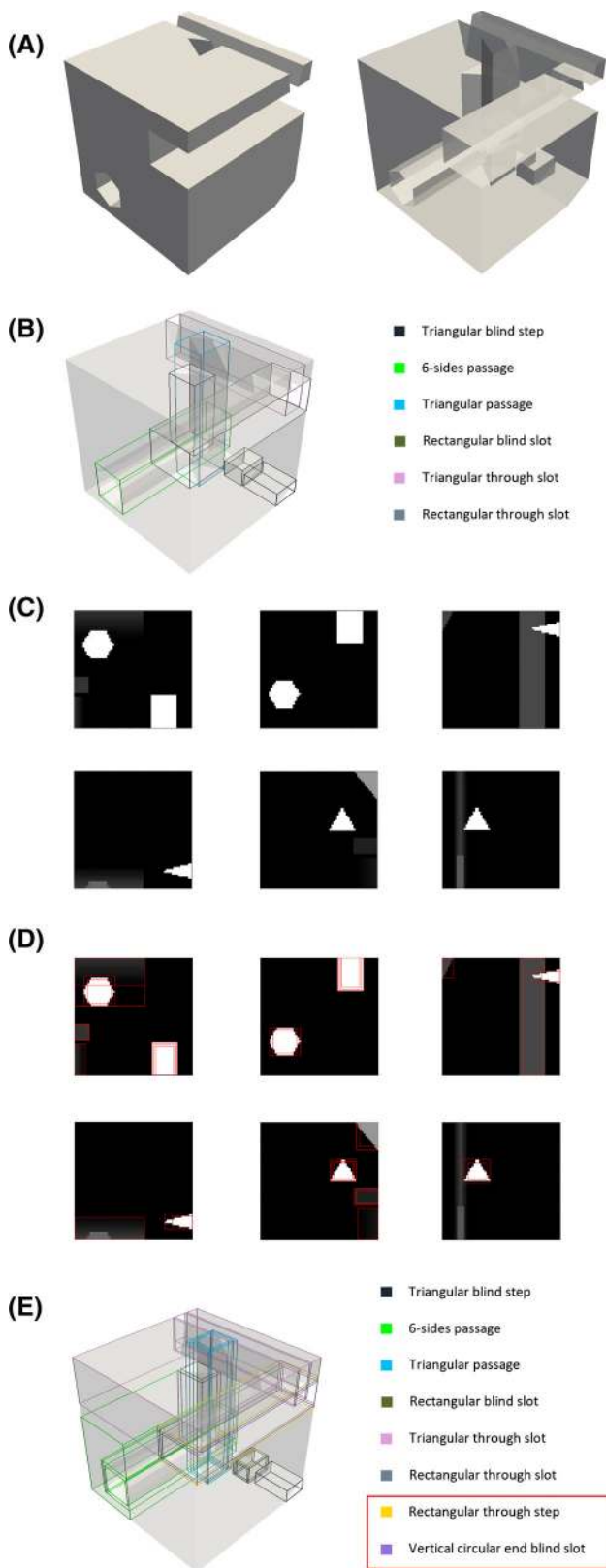
Figure 10 illustrates an example of the whole view-based feature segmentation and recognition process. It is observed from Fig. 10a and b that this 3D model consists of eight features (one 6-sides passage, one triangular through slot, one triangular passage, two triangular blind steps, one rectangular blind slot, and two rectangular through slots). First, its six view images are captured accordingly. Then, selective search algorithm is applied to each view image to achieve all possible feature locations, which are highlighted with red bounding boxes in Fig. 10d. Finally, the corresponding 3D features are extracted from this 3D model, and employed as the input the MsvNet for recognition. A large number of features are recognised accordingly (see Fig. 10e).

### Proposal selection

It is observed from Fig. 10e that a large number of redundant features (e.g. 6-sides passages) and wrongly segmented features (e.g. vertical circular end blind slot) have been recognised. To tackle this problem, soft non-maximum suppression algorithm (Bodla et al. 2017), the state-of-the-art proposal selection algorithm, is adopted as it is capable of identifying object regions from a large number of region proposals effectively.

Algorithm 2 illustrates how soft non-maximum suppression (Bodla et al. 2017) works. This algorithm first starts with a set of 3D feature region proposals  $B$ , and corresponding recognition scores  $R_s$ . In this approach, a feature region





**Fig. 10** An example of the view-based feature segmentation and recognition process: **a** original 3D model, **b** ground truth, **c** view images achieved from six directions, **d** selective search results, **e** identified features (wrongly segmented features are highlighted with a red box) (Color figure online)

### Algorithm 2: Soft non-maximum suppression

**Input:**  $B = \{b_1, \dots, b_n\}$ ,  $Rs = \{rs_1, \dots, rs_n\}$ ,  $B$  is a set of 3D feature region proposals,  $Rs$  contains corresponding recognition scores

**Output:** a new 3D feature region proposal set  $D$

$D = \emptyset$

```

while  $B \neq \emptyset$  do
   $m = \operatorname{argmax} Rs$ 
  Move  $b_m$  from  $B$  to  $D$ 
  for  $b_i \in B$  do
    |  $rs_i = rs_i f(b_i, b_m)$ 
  end
end
Update  $D$  according to  $Rs$ 
return  $D$ 

```

proposal is represented as a 3D bounding box, while the recognition score of each feature is achieved from the softmax layer of the MsvNet. Then, a greedy process is conducted to select a feature region proposal with highest recognition score from  $B$ , and reduce the scores of the rest of proposals in  $B$  proportional to the Intersection over Union (IoU) value, which is calculated as

$$iou = \frac{\text{area of overlap}}{\text{area of union}} \quad (7)$$

This process iterates until all the region proposals in  $B$  are moved to a new set  $D$ . Finally, the region proposals in  $D$  with high recognition scores will be selected via the Maximum Cut algorithm (Largeron et al. 2012). As exemplified in Fig. 11, those redundant and wrongly recognised features are eliminated via this algorithm.

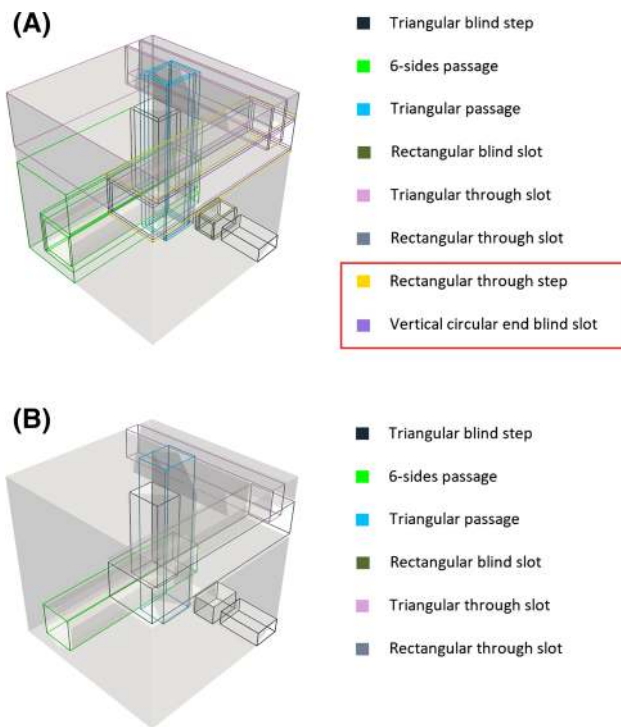
## Results for single feature recognition

Based on the machine learning method proposed in “Single feature recognition” section, this section compares the MsvNet to the FeatureNet (Zhang et al. 2018) in terms of recognition accuracy and efficiency, examines the benefits of different learning strategies in the MsvNet for single feature recognition, and presents a detailed user guide to using the MsvNet in practice.

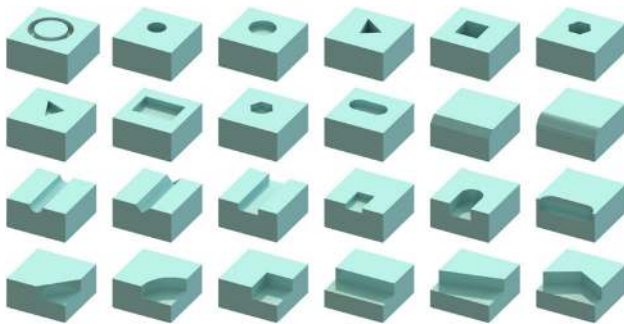
All the experiments in this section are conducted on a PC with Intel Core i9-9900X CPU, 128 GB memory and NVIDIA GeForce RTX 2080 TI GPU.

## Benchmark dataset

In this experiment, a benchmark dataset is required to examine the benefits of the MsvNet, and make a comparison between different approaches. Thus, the dataset used in FeatureNet (Zhang et al. 2018) is employed as it is a very large benchmark set with 24 commonly occurring features, e.g.



**Fig. 11** Proposal selection process: **a** recognised features achieved from the previous section, and **b** final results after the proposal selection process



**Fig. 12** 24 features in the benchmark dataset (Zhang et al. 2018)

hole, ring, slot, pocket and step (see Fig. 12). For each type of feature, there are 1000 randomly generated 3D models with six orientations. In total, this set contains 144,000 ( $= 24 \times 1000 \times 6$ ) different samples which are stored as a stereolithography (STL) file format.

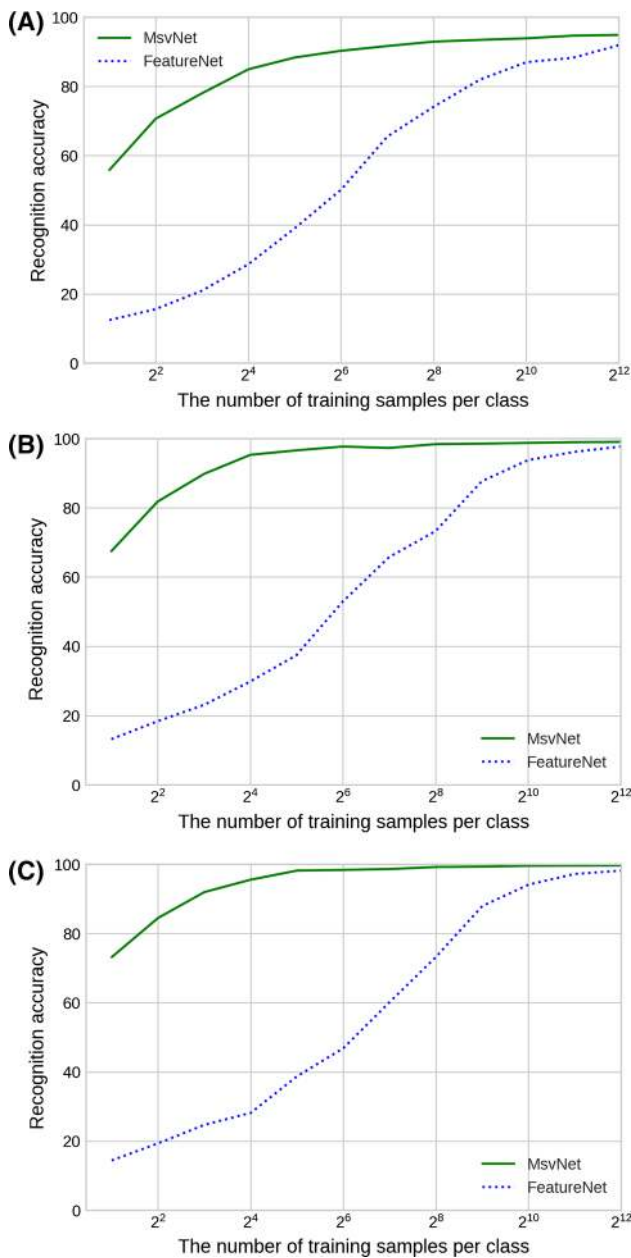
Since both the MsvNet and FeatureNet require 3D voxel grid information for feature recognition, a library named *binvox* (Min 2004) is employed to convert STL models to voxelised models as suggested by Zhang et al. (2018). Three different grid resolutions ( $16 \times 16 \times 16$ ,  $32 \times 32 \times 32$ ,  $64 \times 64 \times 64$ ) are used in this experiment for the purpose of comparison. For each resolution, the data set is split into a training set (80% of the entire set), a validation set (10% of the set) and a test set (10% of the set) as suggested by Ng (2017).

## Recognition accuracy

In the MsvNet, the numbers of training epochs at two stages are set as 20 and 100, respectively. The batch size is set as 64, the learning rate is set as 0.0001, and the number of views is 12 as suggested by Su et al. (2015). All the view images are resized to  $3 \times 64 \times 64$  as this input size constitutes a good tradeoff between the recognition efficiency and accuracy. In the FeatureNet, the number of training epochs is set as 100 to fully train the neural network.<sup>4</sup> The rest parameters are set to defaults as suggested by Zhang et al. (2018). Both the MsvNet and FeatureNet are evaluated at three different grid resolutions. To examine the generalisation performances of the two methods with different numbers of training samples,  $N_s$  training samples are randomly selected from each feature for training. For instance, when  $N_s = 2$ , the total number of samples employed for training is 48 ( $= 2 \times 24$ ). In this experiment,  $N_s \in \{2, 2^2, \dots, 2^{12}\}$ .

Figure 13 illustrates test accuracies of the two approaches with different numbers of training samples at three resolutions. It is observed from this figure that the MsvNet achieves a better generalisation performance than the FeatureNet across all training set sizes. At the resolution of  $16 \times 16 \times 16$ , the optimal accuracies of the MsvNet and the FeatureNet are 94.88% and 91.91% respectively when using 4096 ( $= 2^{12}$ ) samples per class for training. When using 2 training samples per class, the MsvNet achieves 55.86% accuracy, while the FeatureNet obtains 12.47% accuracy. The MsvNet achieves a near-optimal performance (92.94%) when using 256 ( $= 2^8$ ) training samples per class, which is fewer than the FeatureNet (4096 samples per class). At the resolution of  $32 \times 32 \times 32$ , the optimal accuracies of the MsvNet and FeatureNet are 99.06% and 97.72% respectively when using 4096 training samples per class. The MsvNet achieves 67.46% accuracy using 2 training samples per class, while the FeatureNet obtains 13.24% accuracy. The MsvNet achieves a near-optimal performance (97.71%) when using 64 ( $= 2^6$ ) training samples per class, which is fewer than the FeatureNet (4096 samples per class). At the resolution of  $64 \times 64 \times 64$ , the optimal accuracies of the MsvNet and the FeatureNet are 99.67% and 98.17% respectively using 4096 training samples. When using 2 training samples per class, the MsvNet achieves 73.17% accuracy, while the FeatureNet obtains 14.42% accuracy. The MsvNet achieves a near-optimal performance (97.23%) using 32 ( $= 2^5$ ) training samples per

<sup>4</sup> It is worth nothing that the paper employs the term *training epoch* instead of the term *training step* used by Zhang et al. (2018). One training epoch means the entire dataset is passed through the network for training, while one training step means one min-batch samples is passed through the network for training.



**Fig. 13** Experiment I: test accuracy with different number of training samples at the resolution **a** 16 × 16 × 16, **b** 32 × 32 × 32, and **c** 64 × 64 × 64

class, which is fewer than the FeatureNet (4096 samples per class). The aforementioned results are summarised in Table 2.

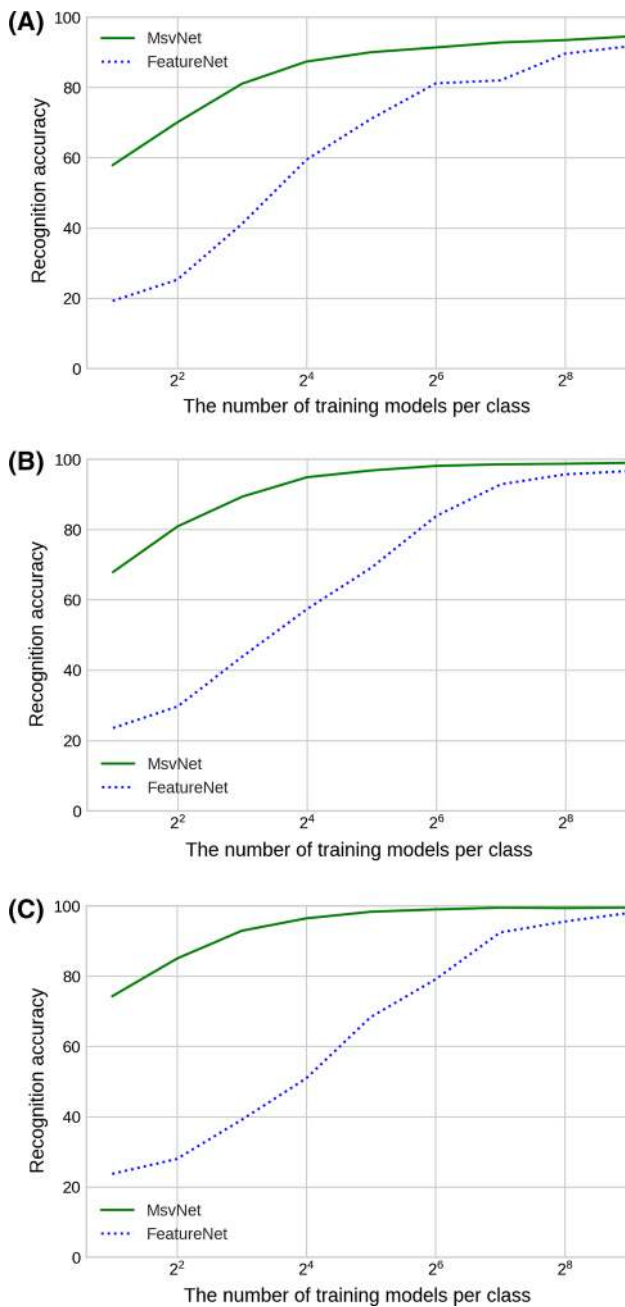
**Table 2** Experiment I: recognition performance (accuracy and the number of training samples per class) of the two approaches at three different resolutions

|              | Optimal performance |               | Near-optimal performance |               |
|--------------|---------------------|---------------|--------------------------|---------------|
|              | MsvNet              | FeatureNet    | MsvNet                   | FeatureNet    |
| 16 × 16 × 16 | 94.88% (4096)       | 91.91% (4096) | 92.94% (256)             | 88.25% (2048) |
| 32 × 32 × 32 | 99.06% (4096)       | 97.72% (4096) | 97.71% (64)              | 96.18% (2048) |
| 64 × 64 × 64 | 99.67% (4096)       | 98.17% (4096) | 98.19% (32)              | 97.23% (2048) |

A second experiment is conducted to further examine the benefits of the two approaches. In this experiment,  $N_m$  voxelised models are selected from each type of feature. For each voxelised model, its six orientations are employed for training. For instance, when  $N_m = 2$ , the total number of samples employed for training is 288 ( $= 2 \times 6 \times 24$ ). In this experiment,  $N_m \in \{2, 2^2, \dots, 2^9\}$ . The rest experimental settings are identical to those used in the previous experiment.

Figure 14 illustrates test accuracies of the two approaches with different numbers of 3D models at three resolutions. It is evident from this figure that the MsvNet achieves a better generalisation performance than the FeatureNet across all training set sizes. At the resolution of 16 × 16 × 16, the optimal accuracies of the MsvNet and the FeatureNet are 94.51% and 91.74% respectively when using 512 ( $= 2^9$ ) models per class for training. The optimal accuracies of the two methods become 57.88% and 19.23% respectively when using 2 models per class. The MsvNet achieves a near-optimal performance (92.79%) when using 128 ( $= 2^7$ ) models per class, which is fewer than the FeatureNet with 512 models per class. At the resolution of 32 × 32 × 32, the optimal accuracies of the MsvNet and the FeatureNet are 98.86% and 96.65% respectively when using 512 3D models per class, and become 67.89% and 23.58% respectively when using 2 3D models per class. The MsvNet achieves a near-optimal performance (96.81%) when using 32 ( $= 2^5$ ) 3D models per class for training, which is fewer than the FeatureNet with 256 models per class. At the resolution of 64 × 64 × 64, the optimal accuracies of the MsvNet and the FeatureNet are 99.51% and 97.99% respectively using 512 ( $= 2^9$ ) training samples, and then become 74.28% and 23.72% respectively when using 2 training models per class. The MsvNet achieves a near-optimal performance (98.33%) using 32 ( $= 2^5$ ) training samples per class, which is fewer than the FeatureNet (256 models per class). The aforementioned results are summarised in Table 3.

In summary, the comparative study demonstrates that the proposed method works well in single feature recognition. In particular, it is capable of achieving the state-of-the-art performance on the benchmark dataset with only a small number of training samples.



**Fig. 14** Experiment II: test accuracy with different number of training models at the resolution **a**  $16 \times 16 \times 16$ , **b**  $32 \times 32 \times 32$ , and **c**  $64 \times 64 \times 64$

### Training and testing efficiency

This section further makes a comparison between the MsvNet and the FeatureNet in terms of efficiency as it is essential for machine learning and real-time feature recognition. In general, efficiency can be measured by the time taken in the training/testing process.

Table 4 shows the training time taken by the two approaches at three different resolutions. When using 4096

samples per class for training, the optimal recognition performances are achieved. In this situation, the FeatureNet is more efficient than the proposed method at the resolution of  $16 \times 16 \times 16$  and  $32 \times 32 \times 32$ , but less efficient at the resolution of  $64 \times 64 \times 64$ . In the near-optimal situation, the proposed method is more efficient than the FeatureNet at the resolution of  $32 \times 32 \times 32$  and  $64 \times 64 \times 64$  since a small number of training samples are required.<sup>5</sup>

Table 5 illustrates the test time taken by the two approaches in recognising a single sample at three resolutions. The MsvNet is less efficient when a  $16 \times 16 \times 16$  or  $32 \times 32 \times 32$  3D model is used, but more efficient at the resolution of  $64 \times 64 \times 64$ . It is notable that both methods have satisfactory runtime performance, which allows for real-time feature recognition.

### Effects of different learning strategies

As discussed in “Single feature recognition” section, the proposed approach adopts a number of machine learning techniques (e.g. multiple sectional view representation, data augmentation, transfer learning) to construct a reliable classifier. It is essential to validate the performances of these techniques. For instance, it is anticipated that transfer learning improves the target task performances rather than degrade them. Thus, this section further examines the benefits of different training strategies from the perspective of feature recognition. In this experiment, the benchmark dataset described in the previous section is also employed. Model resolution is set as  $64 \times 64 \times 64$ , while the number of training samples per class is set as 32. Under this setting, a near-optimal performance can be achieved with only a few training samples.

For a thorough evaluation, a number of experiments are conducted based on the following settings:

- (1) The MsvNet with 12 sectional views is employed in this setting. A pre-trained model on the ImageNet is adopted to initialise the parameters in the neural network, and data augmentation is enabled across all the training stages. At the stage 1, the MsvNet takes single views of 3D models to fine-tune the parameters in the network. This is the default configuration employed in the previous sections.
- (2) Three sectional views are adopted during training and testing. The rest configurations are identical to those in the previous setting.

<sup>5</sup> Note that the number of training epochs in the FeatureNet is set as 100 in the paper, which leads to a higher recognition accuracy but a longer training time than the experiments conducted by Zhang et al. (2018).

**Table 3** Experiment II: recognition performance (accuracy and the number of training models per class) of the two approaches at three different resolutions

|                          | Optimal performance |              | Near-optimal performance |              |
|--------------------------|---------------------|--------------|--------------------------|--------------|
|                          | MsvNet              | FeatureNet   | MsvNet                   | FeatureNet   |
| $16 \times 16 \times 16$ | 94.51% (512)        | 91.74% (512) | 92.79% (128)             | 89.60% (256) |
| $32 \times 32 \times 32$ | 98.86% (512)        | 96.65% (512) | 96.81% (64)              | 95.70% (256) |
| $64 \times 64 \times 64$ | 99.51% (512)        | 97.99% (512) | 98.33% (32)              | 95.54% (256) |

**Table 4** Training efficiency (training time and the number of training samples per class) of the two approaches at three different resolutions

|                          | Optimal performance |                    | Near-optimal performance |                    |
|--------------------------|---------------------|--------------------|--------------------------|--------------------|
|                          | MsvNet              | FeatureNet         | MsvNet                   | FeatureNet         |
| $16 \times 16 \times 16$ | 712.35 min (4096)   | 78.70 min (4096)   | 47.93 min (256)          | 40.52 min (2048)   |
| $32 \times 32 \times 32$ | 742.67 min (4096)   | 310.65 min (4096)  | 20.16 min (64)           | 167.93 min (2048)  |
| $64 \times 64 \times 64$ | 871.23 min (4096)   | 2139.55 min (4096) | 19.21 min (32)           | 1046.35 min (2048) |

**Table 5** Testing efficiency of the two approaches at three different resolutions

|            | $16 \times 16 \times 16$<br>(ms) | $32 \times 32 \times 32$<br>(ms) | $64 \times 64 \times 64$<br>(ms) |
|------------|----------------------------------|----------------------------------|----------------------------------|
| MsvNet     | 2.44                             | 2.48                             | 3.75                             |
| FeatureNet | 0.13                             | 0.65                             | 4.86                             |

- (3) The MsvNet with 80 sectional views is employed. Pre-training (or transfer learning), fine-tuning and data augmentation are enabled during training.
- (4) The MsvNet with 12 sectional views is used in this setting. All the learning strategies are enabled except for transfer learning, which means that the parameters in the deep neural network is initialised randomly.
- (5) The MsvNet with 12 sectional views is used in this setting. Transfer learning and data augmentation are enabled, while fine-tuning is disabled during training.
- (6) The MsvNet with 12 sectional views is used in this setting. All the learning strategies are enabled except for data augmentation.
- (7) The MsvNet with 12 sectional views and transfer learning is adopted.
- (8) The MsvNet with 12 sectional views is employed. Fine-tuning is enabled at the training stage 1, while transfer learning and data augmentation are disabled.
- (9) The MsvNet with 12 sectional views and data augmentation is used.
- (10) The MsvNet with 12 sectional views is adopted. Transfer learning, fine-tuning and data augmentation are disabled.
- (11) The FeatureNet is employed as a baseline for the comparison purpose.

Table 6 shows the classification results under different experimental settings. It is observed from the setting (1), (2)

and (3) that the MsvNet with 12 views produces an optimal performance than the networks with 3 and 80 views. From the setting (1) and (4), it is notable that transfer learning considerably increases the recognition accuracy by 5.09% (93.10%→98.19%). This phenomenon is more observable from the results achieved from the setting (7) and (10), where transfer learning increases the accuracy by 20.73% (72.96%→93.69%). This results indicate that this transfer is suitable for this research problem. It is also evident from the setting (1) and (5) that fine-tuning slightly increases the learning performance by 0.83% (97.36%→98.19%). From the setting (6) and (7), it is interesting to note that fine-tuning does not increase the accuracy. During the fine-tuning, single view images are adopted to fine-tune the model. At this stage, the input of the model is a single view image, while the output is its label. In some situations, a single random sectional view image may contain insufficient yet misleading information about the feature. For instance, a sectional view image for a blind hole might be identical to a sectional view image for a through hold. Thus, using these single view images as training examples to fine-tune the model will not always lead to a better result. From the setting (1) and (6), it is observed that data augmentation considerably increases the recognition accuracy by 5.03% (93.16%→98.19%). This phenomenon is more observable from the results achieved from the setting (9) and (10), where data augmentation increases the accuracy by 20.20% (72.96%→93.16%). From the setting (10) and (11), it is visible that a 2D CNN with view-based representation outperforms 3D CNN with voxel-based representation in terms of recognition accuracy.

In summary, view-based representation, transfer learning and data augmentation play an important role in the MsvNet for feature recognition as these learning strategies can increase the accuracy by a considerable amount.

**Table 6** Single feature recognition results on the benchmark set under different experimental settings

|      | Method     | Number of views | Pre-training | Fine-tuning | Data augmentation | Accuracy |
|------|------------|-----------------|--------------|-------------|-------------------|----------|
| (1)  | MsvNet     | 12              | ✓            | ✓           | ✓                 | 98.19%   |
| (2)  | MsvNet     | 3               | ✓            | ✓           | ✓                 | 90.16%   |
| (3)  | MsvNet     | 80              | ✓            | ✓           | ✓                 | 97.91%   |
| (4)  | MsvNet     | 12              |              | ✓           | ✓                 | 93.10%   |
| (5)  | MsvNet     | 12              | ✓            |             | ✓                 | 97.36%   |
| (6)  | MsvNet     | 12              | ✓            | ✓           |                   | 93.16%   |
| (7)  | MsvNet     | 12              | ✓            |             |                   | 93.69%   |
| (8)  | MsvNet     | 12              |              | ✓           |                   | 74.60%   |
| (9)  | MsvNet     | 12              |              |             | ✓                 | 93.19%   |
| (10) | MsvNet     | 12              |              |             |                   | 72.96%   |
| (11) | FeatureNet | –               | –            | –           | –                 | 38.72%   |

## A practical guide

As shown in above experiments, 20% of samples (1200 samples for each feature,  $1200 \times 24$  in total) are employed for validation and testing purposes. In general, a large test/validation size can lead to an accurate estimation of classification accuracy, but requires a huge amount of human work for data annotation. Thus, this section presents a practical guide to show how to choose a small test and validation set sizes that can retain all properties underlying the whole data population from the perspective of statistics.

Suppose that a test set consists of  $n$  samples. Let  $p_i$  denote whether the  $i$ th sample in the test set is correctly classified.  $p_i = 0$  when this sample is misclassified, and is 1 otherwise. It is observed that the random variable  $p_i$  is subject to a Bernoulli distribution. It equals value 1 with a probability  $\mu_p$ . This probability is the real classification accuracy on the whole population. Thus, the classification accuracy on the test set can be calculated as

$$\bar{p} = \frac{1}{n} \sum_{i=1}^n p_i \quad (8)$$

It is expected that the classification accuracy on the test set ( $\bar{p}$ ) is close to the real classification accuracy on the whole population ( $\mu_p$ ). According to the central limit theorem,  $\bar{p}$  is subject to a normal distribution when the test set is sufficiently large (e.g.  $n \geq 30$ ). Thus, a 95% CI for  $\mu_p$  can be calculated as

$$\bar{p} \pm \frac{1.96\delta}{\sqrt{n}} \quad (9)$$

where  $\delta$  refers to population standard deviation. Since  $p_i$  is subject to a Bernoulli distribution, the population standard deviation can be calculated by

$$\delta = \sqrt{\mu_p * (1 - \mu_p)} \quad (10)$$

The proper test set size,  $n$ , can be estimated by

$$n = \frac{1.96^2 \delta^2}{d^2} = \frac{1.96^2 \mu_p (1 - \mu_p)}{d^2} \quad (11)$$

where  $d$  refers to a maximum allowable difference between the estimated accuracy and the real accuracy.

Take the case in the paper as an example, the user could set  $d$  as 3%, and assume that  $\mu_p \geq 85\%$ . The necessary test/validation set size can be calculated as follow:

$$n \leq \frac{1.96^2 * 0.85 * (1 - 0.85)}{0.03^2} \approx 545 \quad (12)$$

For each type of features, about 23 ( $\approx 545/24$ ) samples are required for testing and validation.

In practice, a user could set the training set, validation set and test set sizes for each feature as 8, 23, and 23, respectively. Then, the user trains the deep network with different hyper-parameters (e.g. number of views, learning rates, number of epochs) on the training data set, and observes the performances on the validation set. For instance, the experiments conducted in the previous section employ 12 views for recognition. If the feature is very complex, 12 views may not be sufficient. In this case, an optimal value of this hyper-parameter is selected according to the validation performances. Once optimal hyper-parameters are found, training and validation sets can be merged together to form a new set. Finally, the user retrain the deep network on the new set, and test the final model on the test set. In total, only 54 samples for each feature are required for training and testing. This simple strategy can further minimise the number of annotated samples.

**Table 7** Benchmark dataset generation setting

| 3D model index | Number of features | Parameter scale |
|----------------|--------------------|-----------------|
| 0–99           | 2                  | 1               |
| 100–199        | 2                  | 1/2             |
| 200–299        | 3                  | 1/2             |
| 300–399        | 4                  | 1/2             |
| 400–499        | 5                  | 1/3             |
| 500–599        | 6                  | 1/3             |
| 600–699        | 7                  | 1/3             |
| 700–799        | 8                  | 1/4             |
| 800–899        | 9                  | 1/4             |
| 900–999        | 10                 | 1/4             |

## Results for multi-feature recognition

Based on the machine learning method proposed in “Multi-feature recognition” section, this section compares the MsvNet to the FeatureNet (Zhang et al. 2018) for multi-feature recognition. All the experiments in this section are conducted on the same PC employed in the previous section.

The dataset and source code that produces the results reported in this section are available online.<sup>6</sup>

### Benchmark dataset

In this experiment, a large benchmark dataset is required to make a comparison between different approaches to multi-feature recognition. Thus, a synthetic dataset that consists of 1000 STL models with multiple features is constructed via the Solidworks API.

In this set, 24 machining features suggested by Zhang et al. (2018) are employed to construct 3D models, and all the models are built on  $10 \times 10 \times 10 \text{ cm}^3$  raw stocks. Table 7 shows the detailed information about this benchmark dataset. When creating a 3D model, a fixed number of features (2–10) are randomly placed on the stock. The range of parameters (e.g. feature coordinates and sizes) used for creating the first 100 models are identical to those in the FeatureNet dataset (Zhang et al. 2018). When creating the following 900 models (model 100–999), the maximal sizes of features (e.g. width and height of a feature) are set as 1/2, 1/2, 1/2, 1/3, 1/3, 1/3, 1/4, 1/4, 1/4 of sizes used in the FeatureNet dataset (Zhang et al. 2018). This setting would allow for creating 3D models with a large number of features.

Each multi-feature model has a degree of overlap. To better examine the performances of the proposed algorithm under different degrees of overlap, the benchmark dataset is equally divided into ten groups according to the IoU value

of each 3D model. As exemplified in Fig. 15, the models in the group 1–3 contain slightly overlapped features, while the models in the group 4–8 consist of highly overlapped features. The models in the group 9–10, however, contains extremely high intersecting features, where some faces of a feature are removed by other features.<sup>7</sup> One thing that needs to be noticed is that some models in this benchmark set might appear to be unusual from a normal engineering point of view, since a random algorithm is adopted to place the features. However, it will not affect the comparison since this set consists of a large number of complicated models with highly intersecting features.

### Recognition performance

This experiment aims to make a fair comparison between the MsvNet and the FeatureNet for multi-feature recognition. Therefore, both neural networks are trained on the FeatureNet dataset provided by Zhang et al. (2018), and tested on the multi-feature dataset constructed by the authors of this paper.

In the training phase, the grid resolution of training samples is set as  $64 \times 64 \times 64$ . For each voxelised model, its 24 orientations (instead of six orientations) are employed for training to make the methods orientation invariant. The rest experimental settings are identical to those optimal settings used in the “Recognition accuracy” section. The above settings can guarantee that both the MsvNet and the FeatureNet are trained optimally.

In the testing phase, the deep learning models trained at the previous stage are applied to the multi-feature dataset constructed in this paper. Since both the MsvNet and FeatureNet require 3D voxel grid information for feature recognition, the *binvox* library is also adopted to voxelise models into  $64 \times 64 \times 64$  grids. As the source code of the FeatureNet for multi-feature recognition is unavailable online, the authors of this paper reimplemented the multi-feature recognition part of the FeatureNet according to their original paper (Zhang et al. 2018). In this reimplemented version, the disconnected machining features are separated via the connected component labelling algorithm, while the connected ones are separated via the watershed algorithm (under the default setting) in scikit-image package of python as suggested by Zhang et al. (2018).

In this experiment, two popular machine learning evaluation metrics named *precision* and *recall* are adopted to measure the multi-feature recognition performances. These two metrics are chosen since they are suitable for multi-class classification problem (Sokolova and Lapalme 2009). The precision score for the feature type  $i$  is calculated as

<sup>6</sup> <https://github.com/PeizhiShi/MsvNet>.

<sup>7</sup> Please note that the data groups are determined according to the degree of overlap rather than the number of features appeared.

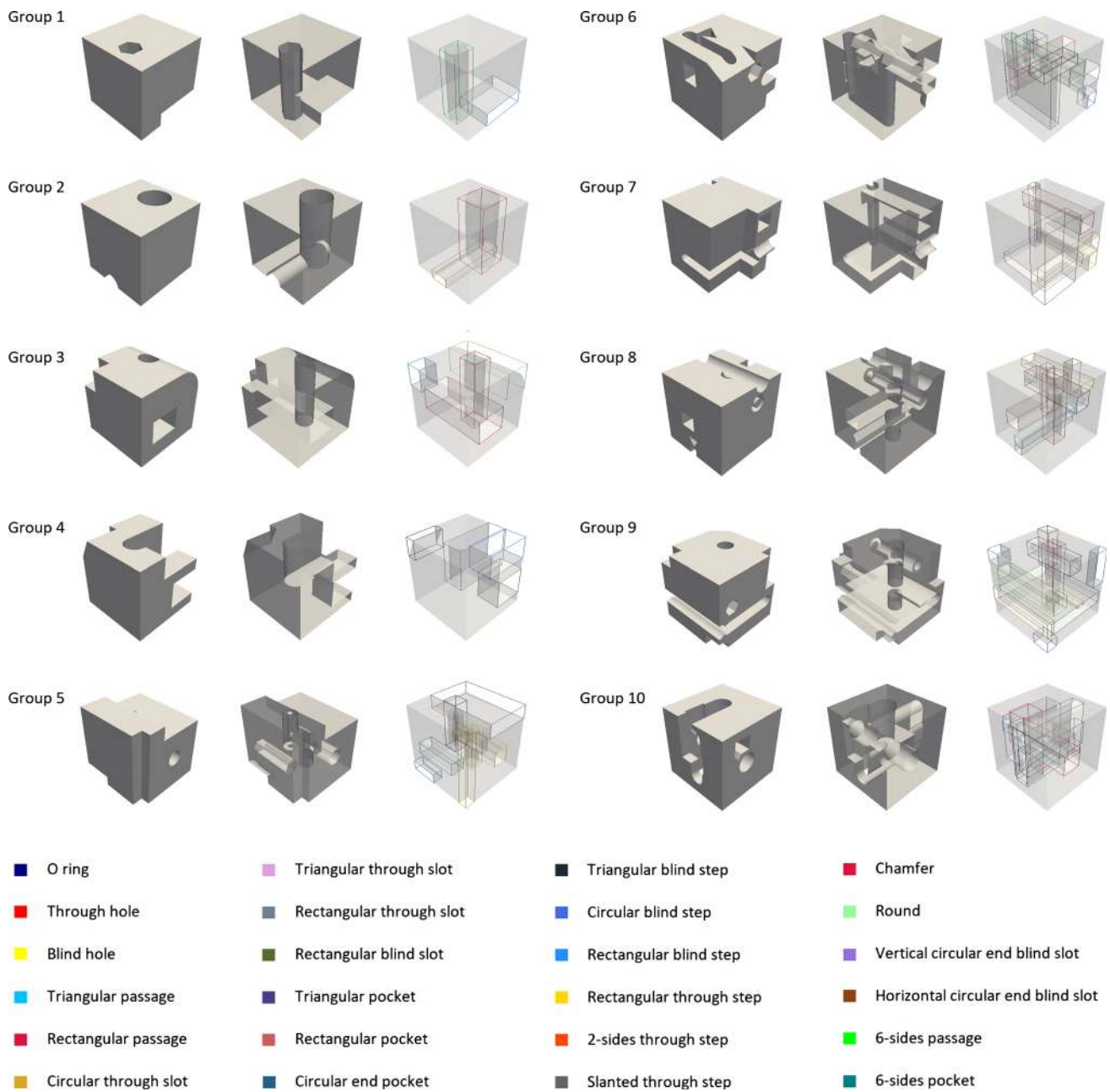


Fig. 15 Benchmark dataset for multi-feature recognition

$$precision_i = \frac{tp_i}{tp_i + fp_i} \quad (13)$$

where  $tp_i$  refers to the number of correctly identified type  $i$  features, and  $fp_i$  refers to the number of features that are misclassified as type  $i$ . The final precision score is the average of  $precision_i$  values. This metric measures the proportion of predicted features which are correctly predicted. For instance, a 0.9 precision value implies that on average 90% predicted features are correct. The recall score for the feature type  $i$  is calculated as

$$recall_i = \frac{tp_i}{tp_i + fn_i} \quad (14)$$

where  $fn_i$  refers to the number of type  $i$  features that are not identified correctly. The final recall score is the average of  $recall_i$  values. This metric measures the ability of an intelligent system to find all features appeared in 3D models. For instance, a 0.85 recall value means that on average 85% features appeared in 3D models are found correctly.



**Table 8** Multi-feature recognition results (precision) achieved from different groups of the benchmark set

|            | $p^1$ (%) | $p^2$ (%) | $p^3$ (%) | $p^4$ (%) | $p^5$ (%) | $p^6$ (%) | $p^7$ (%) | $p^8$ (%) | $p^9$ (%) | $p^{10}$ (%) |
|------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--------------|
| MsvNet     | 94.93     | 87.52     | 85.54     | 80.57     | 80.77     | 80.87     | 83.37     | 81.05     | 77.19     | 74.69        |
| FeatureNet | 93.10     | 79.63     | 69.14     | 61.20     | 70.27     | 63.21     | 63.16     | 59.90     | 60.19     | 56.18        |

**Table 9** Multi-feature recognition results (recall) achieved from different groups of the benchmark set

|            | $r^1$  | $r^2$  | $r^3$  | $r^4$  | $r^5$  | $r^6$  | $r^7$  | $r^8$  | $r^9$  | $r^{10}$ |
|------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|
| MsvNet     | 96.96% | 88.39% | 82.26% | 76.00% | 72.35% | 73.16% | 72.39% | 70.04% | 65.55% | 60.37%   |
| FeatureNet | 92.57% | 82.42% | 60.18% | 57.12% | 52.55% | 51.91% | 48.36% | 48.82% | 43.02% | 39.65%   |

**Table 10** Testing efficiency of the two approaches to multi-feature recognition

|              | MsvNet    | FeatureNet |
|--------------|-----------|------------|
| Average time | 806.67 ms | 426.99 ms  |

Tables 8 and 9 illustrate the multi-feature recognition results, where  $p^i$  (or  $r^i$ ) refers to the precision (or recall) scores achieved from the  $i$ th group. It is observed that the MsvNet outperforms the FeatureNet in terms of precision and recall across all data groups. From Table 8, it is evident that the MsvNet achieves high precision scores across almost all the groups, which indicates that this method produces substantially more correct results than incorrect ones. Table 9 suggests that the MsvNet achieves higher recall scores when recognising models in the group 1–8, which indicates that this method finds most of the features appeared in the 3D models. The models in the rest two groups, however, are composed of extremely high intersecting features. In some situations, most faces of a feature are removed due to the feature intersections. Thus, identifying features according to these incomplete faces is rather challenging (Han et al. 2000), which results in low recall scores.

### Testing efficiency

This section further makes a comparison between the MsvNet and the FeatureNet in terms of efficiency for multi-feature recognition.

Table 10 illustrates the average recognition test time taken by the two approaches in recognising a single sample at the proposed benchmark set. It is observed that the MsvNet is less efficient than the FeatureNet for multi-feature recognition since the MsvNet demands recognising a large number of redundant features, which could slow down the recognition process.

### Conclusion

The paper presents a novel learning-based technique for single and multi-feature recognition. A deep learning framework based on multiple sectional view representation named MsvNet is proposed. A set of experiment were conducted to compare the proposed MsvNet and the FeatureNet with respect to recognition accuracy and efficiency for both single and multi-feature recognition. Experimental results suggest that (a) the proposed method can achieve the state-of-the-art performances with only a few training samples (8–32 samples per class) for single feature recognition; (b) the learning strategies employed in the MsvNet (e.g. data augmentation, transfer learning, multiple sectional view representation) is well capable of enhancing the learning performances on a greater scale; (c) the proposed multi-feature recognition algorithm outperforms the state-of-the-art learning-based multi-feature recognition in terms of recognition performances.

A possible future work is to keep improving the multi-feature recognition performance. Therefore, advanced machine learning techniques will be required. The second possible work is to tackle the free-form multi-feature recognition problem, which cannot be directly solved by the proposed method. Further, studies can be conducted for the generation of manufacturing information from the 3D models, for example, direction of features, geometrical and topological information of features.

**Acknowledgements** This research was supported by the EPSRC Future Advanced Metrology Hub (Ref. EP/P006930/1), EPSRC UKRI Innovation Fellowship (Ref. EP/S001328/1), and EPSRC Fellowship in Manufacturing (Ref. EP/R024162/1). The authors would like to thank two anonymous reviewers for their valuable comments and suggestions. The authors would like to thank Zhibo Zhang for providing the dataset and source code of FeatureNet on Github used in this research.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indi-

cate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Al-wswasi, M., & Ivanov, A. (2019). A novel and smart interactive feature recognition system for rotational parts using a step file. *The International Journal of Advanced Manufacturing Technology*, 104, 1–24.
- Babic, B., Nestic, N., & Miljkovic, Z. (2008). A review of automated feature recognition with rule-based pattern recognition. *Computers in Industry*, 59(4), 321–337.
- Babić, B. R., Nešić, N., & Miljković, Z. (2011). Automatic feature recognition using artificial neural networks to integrate design and manufacturing: Review of automatic feature recognition systems. *AI EDAM*, 25(3), 289–304.
- Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Soft-nms—improving object detection with one line of code. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5561–5569.
- Brousseau, E., Dimov, S., & Setchi, R. (2008). Knowledge acquisition techniques for feature recognition in cad models. *Journal of Intelligent Manufacturing*, 19(1), 21–32.
- Campana, G., & Mele, M. (2018). An application to stereolithography of a feature recognition algorithm for manufacturability evaluation. *Journal of Intelligent Manufacturing*. 1–16.
- Ding, L., & Yue, Y. (2004). Novel ann-based feature recognition incorporating design by features. *Computers in Industry*, 55(2), 197–222.
- Dipper, T., Xu, X., & Klemm, P. (2011). Defining, recognizing and representing feature interactions in a feature-based data model. *Robotics and Computer-Integrated Manufacturing*, 27(1), 101–114.
- Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), 167–181.
- Gao, S. (1998). A survey of automatic feature recognition. *Chinese Journal of Computers*, 21, 281–288.
- Gao, S., & Shah, J. J. (1998). Automatic recognition of interacting machining features based on minimal condition subgraph. *Computer-Aided Design*, 30(9), 727–739.
- Han, J., & Requicha, A. A. (1997). Integration of feature based design and feature recognition. *Computer-Aided Design*, 29(5), 393–403.
- Han, J., & Requicha, A. A. (1998). Feature recognition from CAD models. *IEEE Computer Graphics and Applications*, 18(2), 80–94.
- Han, J., Pratt, M., & Regli, W. C. (2000). Manufacturing feature recognition from solid models: A status report. *IEEE Transactions on Robotics and Automation*, 16(6), 782–796.
- Han, J., Kang, M., & Choi, H. (2001a). Step-based feature recognition for manufacturing cost optimization. *Computer-Aided Design*, 33(9), 671–686.
- Han, J. H., Han, I., Lee, E., & Yi, J. (2001). Manufacturing feature recognition toward integration with process planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(3), 373–380.
- Harik, R., Shi, Y., & Baek, S. (2017). Shape terra: mechanical feature recognition based on a persistent heat signature. *Computer-Aided Design and Applications*, 14(2), 206–218.
- Hayasi, M. T., & Asiabanpour, B. (2009). Extraction of manufacturing information from design-by-feature solid model through feature recognition. *The International Journal of Advanced Manufacturing Technology*, 44(11–12), 1191–1203.
- Henderson, M. R., Srinath, G., Stage, R., Walker, K., Regli, W. (1994). Boundary representation-based feature identification. In: *Manufacturing Research and Technology*, vol 20, Elsevier, pp 15–38.
- Huang, Z., & Yip-Hoi, D. (2002). High-level feature recognition using feature relationship graphs. *Computer-Aided Design*, 34(8), 561–582.
- Hwang, J.L. (1992) Applying the perceptron to three-dimensional feature recognition. PhD thesis, Arizona State University.
- Joshi, S., & Chang, T. C. (1988). Graph-based heuristics for recognition of machined features from a 3d solid model. *Computer-Aided Design*, 20(2), 58–66.
- Kannan, T., & Shunmugam, M. (2009). Processing of 3d sheet metal components in step ap-203 format. Part i: Feature recognition system. *International Journal of Production Research*, 47(4), 941–964.
- Kannan, T., & Shunmugam, M. (2009). Processing of 3d sheet metal components in step ap-203 format. Part ii: Feature reasoning system. *International Journal of Production Research*, 47(5), 1287–1308.
- Kao, C.Y. (1993). 3-d manufacturing feature recognition using super relation graph method. In: *Proceedings of the 2nd Industrial Engineering Research Conference*, Publ by IIE, pp. 614–618.
- Largerone, C., Moulin, C., Géry, M. (2012). Mcut: A thresholding strategy for multi-label classification. In: *International Symposium on Intelligent Data Analysis*, Springer, pp 172–183.
- Li, W., Ong, S. K., & Nee, A. Y. (2002). Recognizing manufacturing features from a design-by-feature model. *Computer-Aided Design*, 34(11), 849–868.
- Li, Y., Ding, Y., Mou, W., & Guo, H. (2010). Feature recognition technology for aircraft structural parts based on a holistic attribute adjacency graph. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 224(2), 271–278.
- Lingam, R., Prakash, O., Belk, J., & Reddy, N. (2017). Automatic feature recognition and tool path strategies for enhancing accuracy in double sided incremental forming. *The International Journal of Advanced Manufacturing Technology*, 88(5–8), 1639–1655.
- Lockett, H. L., & Guenov, M. D. (2005). Graph-based feature recognition for injection moulding based on a mid-surface approach. *Computer-Aided Design*, 37(2), 251–262.
- Marchetta, M. G., & Forradellas, R. Q. (2010). An artificial intelligence planning approach to manufacturing feature recognition. *Computer-Aided Design*, 42(3), 248–256.
- Min, P. (2004). Binvx 3d mesh voxelizer. <http://www.patrickmin.com/binvox/>.
- Mokhtar, A., & Xu, X. (2011). Machining precedence of 21/2d interacting features in a feature-based data model. *Journal of Intelligent Manufacturing*, 22(2), 145–161.
- Mokhtar, A., Xu, X., & Lazcanotegui, I. (2009). Dealing with feature interactions for prismatic parts in step-nc. *Journal of Intelligent Manufacturing*, 20(4), 431.
- Nezis, K., & Vosniakos, G. (1997). Recognizing 212d shape features using a neural network and heuristics. *Computer-Aided Design*, 29(7), 523–539.
- Ng, A. (2017). Machine learning yearning. <http://www.deeplearning.ai/machine-learning-yearning/>.
- Ong, S. K., Li, W., & Nee, A. Y. (2003). Step-based integration of feature recognition and design-by-feature for manufacturing applications

- in a concurrent engineering environment. *International Journal of Computer Applications in Technology*, 18(1–4), 78–92.
- Onwubolu, G. C. (1999). Manufacturing features recognition using backpropagation neural networks. *Journal of Intelligent Manufacturing*, 10(3–4), 289–299.
- Öztürk, N., & Öztürk, F. (2001). Neural network based non-standard feature recognition to integrate CAD and CAM. *Computers in Industry*, 45(2), 123–135.
- Öztürk, N., & Öztürk, F. (2004). Hybrid neural network and genetic algorithm based machining feature recognition. *Journal of Intelligent Manufacturing*, 15(3), 287–298.
- Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359.
- Prabhakar, S. (1990). An experiment on the use of neural nets in form feature recognition. Ph.D. thesis, Arizona State University.
- Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5648–5656.
- Rahmani, K., & Arezoo, B. (2007). A hybrid hint-based and graph-based framework for recognition of interacting milling features. *Computers in Industry*, 58(4), 304–312.
- Rameshbabu, V., & Shunmugam, M. (2009). Hybrid feature recognition method for setup planning from step ap-203. *Robotics and Computer-Integrated Manufacturing*, 25(2), 393–408.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint [arXiv:1409.1556](https://arxiv.org/abs/1409.1556).
- Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427–437.
- Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 945–953.
- Su, J.C., Gadelha, M., Wang, R., & Maji, S. (2018). A deeper look at 3d shape classifiers. In: *European Conference on Computer Vision*, Springer, pp. 645–661.
- Sundararajan, V., & Wright, P. K. (2004). Volumetric feature recognition for machining components with freeform surfaces. *Computer-Aided Design*, 36(1), 11–25.
- Sung, R. C., Corney, J. R., & Clark, D. E. (2001). Automatic assembly feature recognition and disassembly sequence generation. *Journal of Computing and Information Science in Engineering*, 1(4), 291–299.
- Sunil, V., & Pande, S. (2008). Automatic recognition of features from freeform surface cad models. *Computer-Aided Design*, 40(4), 502–517.
- Sunil, V., & Pande, S. (2009). Automatic recognition of machining features using artificial neural networks. *The International Journal of Advanced Manufacturing Technology*, 41(9–10), 932–947.
- Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2), 154–171.
- Vandenbrande, J. H., & Requiha, A. A. (1993). Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12), 1269–1285.
- Venu, B., & Komma, V. R. (2017). Step-based feature recognition from solid models having non-planar surfaces. *International Journal of Computer Integrated Manufacturing*, 30(10), 1011–1028.
- Venu, B., Komma, V. R., & Srivastava, D. (2018). Step-based feature recognition system for b-spline surface features. *International Journal of Automation and Computing*, 15(4), 500–512.
- Venuvinod, P. K., & Wong, S. (1995). A graph-based expert system approach to geometric feature recognition. *Journal of Intelligent Manufacturing*, 6(3), 155–162.
- Verma, A. K., & Rajotia, S. (2010). A review of machining feature recognition methodologies. *International Journal of Computer Integrated Manufacturing*, 23(4), 353–368.
- Wang, Q., & Yu, X. (2014). Ontology based automatic feature recognition framework. *Computers in Industry*, 65(7), 1041–1052.
- Wang, Q., Jia, N., Breckon, T. (2018). A baseline for multi-label image classification using ensemble deep cnn. arXiv preprint [arXiv:1811.08412](https://arxiv.org/abs/1811.08412).
- Woo, Y. (2003). Fast cell-based decomposition and applications to solid modeling. *Computer-Aided Design*, 35(11), 969–977.
- Xu, S., Anwer, N., & Mehdi-Souzani, C. (2015). Machining feature recognition from in-process model of nc simulation. *Computer-Aided Design and Applications*, 12(4), 383–392.
- Xu, X. (2009). Integrating advanced computer-aided design, manufacturing, and numerical control. Information Science Reference.
- Zhang, X., Nassehi, A., & Newman, S. T. (2014). Feature recognition from cnc part programs for milling operations. *The International Journal of Advanced Manufacturing Technology*, 70(1–4), 397–412.
- Zhang, Y., Luo, X., Zhang, B., & Zhang, S. (2017). Semantic approach to the automatic recognition of machining features. *The International Journal of Advanced Manufacturing Technology*, 89(1–4), 417–437.
- Zhang, Z., Jaiswal, P., & Rai, R. (2018). FeatureNet: Machining feature recognition based on 3d convolution neural network. *Computer-Aided Design*, 101, 12–22.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.